Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

On oversampling imbalanced data with deep conditional generative models

Val Andrei Fajardo [*], David Findlay, Charu Jaiswal, Xinshang Yin, Roshanak Houmanfar, Honglei Xie, Jiaxi Liang, Xichen She, D.B. Emerson

integrate.ai, 480 University Avenue, Toronto, Canada

ARTICLE INFO

Keywords:
Deep generative models
Conditional variational autoencoders
Class imbalance
Oversampling

ABSTRACT

Class imbalanced datasets are common in real-world applications ranging from credit card fraud detection to rare disease diagnosis. Recently, deep generative models have proved successful for an array of machine learning problems such as semi-supervised learning, transfer learning, and recommender systems. However their application to class imbalance situations is limited. In this paper, we consider class conditional variants of generative adversarial networks and variational autoencoders and apply them to the imbalance problem. The main question we seek to answer is whether or not deep conditional generative models can effectively learn the distributions of minority classes so as to produce synthetic observations that ultimately lead to improvements in the performance of a downstream classifier. The numerical results show that this is indeed true and that deep generative models outperform traditional oversampling methods in many circumstances, especially in cases of severe imbalance.

1. Introduction

A dataset is considered imbalanced if one or more classification categories are under-represented. Learning with imbalanced datasets poses many challenges, most notably with classification tasks, where the goal is to train a classifier that accurately predicts the class to which an observation belongs. Training a classifier on an imbalanced dataset typically results in poor minority class prediction accuracy. In extreme cases, such as where there is an imbalance ratio of 100,000: 1, the classifier may predict the majority class for all observations.

Not only is this problem of learning with imbalanced datasets an interesting machine learning problem in its own right, it is also highly relevant in many industries, including healthcare, retail banking, insurance, and telecommunications, where datasets of interest are often severely imbalanced. Examples of such datasets include observations of screened patients used for diagnosing certain diseases, credit card transactions where the goal is to predict whether or not a transaction is fraudulent, and customer usage and interactions data used for identifying customers most likely to churn or be interested in a new set of products. In these cases, it is critical to correctly classify the minority class since incorrectly labelling a sample as a false positive or as a false negative produces negative consequences and business risks.

Methods for learning with imbalanced data fall under two main categories, namely data-level methods and classifier-level methods (Buda, Maki, & Mazurowski, 2018; He, & Garcia, 2008). With data-level techniques, the idea is to first reduce, or remove, the skew of the dataset by either oversampling the minority classes (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) or undersampling those that are in the majority (Haixiang, Yijing, Shang, Mingyun, Yuanyue et al., 2017), and then training a standard classifier on the modified dataset. On the other hand, classifier-level methods do not alter the dataset but instead involve modifications to either the objective function used for training the classifier (Maloof, 2003; McCarthy, Zabar, & Weiss, 2005) or how inference is made from the classifier's outputs (Richard, & Lippmann, 1991; S. Lawrence, Burns, Back, Tsoi, & Giles, 1998).

In this paper, the problem of learning with imbalanced data is addressed via oversampling. More specifically, we set out to learn the conditional distribution of observations using deep generative modelling techniques. In recent years, deep generative models, most notably generative adversarial networks (GANs) (I. Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma, & Welling, 2014), have been applied to a vast array of machine learning applications such as semi-supervised learning (Kingma, Mohamed, Rezende, & Welling, 2014), structured output prediction (Sohn, Lee, & Yan, 2015), transfer learning (Isola, Zhu, Zhou, & Efros, 2017), recommender systems (Liang, Krishnan, Hoffman, & Jebara, 2018), security (Samangouei, Kabkab, & Chellappa, 2018), as well as handling more sensitive issues such as

* Corresponding author.
E-mail addresses: andrei@integrate.ai (V.A. Fajardo), david@integrate.ai (D. Findlay), charu@integrate.ai (C. Jaiswal), xinshang@integrate.ai (X. Yin), roshan@integrate.ai (R. Houmanfar), holly@integrate.ai (H. Xie), garcia@integrate.ai (J. Liang), xichen@integrate.ai (X. She), davidemerson@integrate.ai (D.B. Emerson).

https://doi.org/10.1016/j.eswa.2020.114463
Received 1 August 2019; Received in revised form 4 December 2020; Accepted 4 December 2020
Available online 13 December 2020

fairness in machine learning algorithms (Louizos, Swersky, Li, Welling, & Zemel, 2016). However, the application of deep generative models for oversampling imbalanced datasets is limited. Specifically, at the time of writing, our scan of the literature on this subject uncovered the papers by Mariani, Scheidegger, Istrate, Bekas, and Malossi (2018) and Douzas, and Bacao (2018), both of which consider GAN-inspired oversampling approaches, as well as the paper by Lopez-Martin, Carro, Sanchez-Esguevillas, and Lloret (2017) where a variant of the VAE was utilized for the intrusion detection problem. Therefore, the research in this paper seeks to answer the question of whether deep conditional generative models can be used to synthetically oversample minority classes in a way that ultimately leads to improvements in the performance of a downstream classifier. The main contributions of this paper are described as follows:

- Inspired by the work in Kingma et al. (2014), we propose a conditional VAE that is able to learn class-dependent distributions.
- In contrast to the above mentioned related works, the numerical experiments herein compare the relative performances of various deep conditional generative models to one another.
- We demonstrate, through numerical examples, that deep conditional generative models, especially conditional VAEs, provide the ability to balance datasets ultimately leading to more accurate classifiers.

The remainder of the paper is organized as follows. In Section 2, we provide a brief review of deep generative models. In Section 3, we discuss conditional variants of these deep generative models and explain how these techniques can be used for the imbalance problem. The performance of these new methods is illustrated through two numerical experiments in Section 4. Finally, in Section 5, we offer some concluding remarks as well as thoughts on potential future work in this area.

2. Review of deep generative models

Consider a dataset of observations $\mathbf{X} = \{x_i, i = 1, 2, \dots, n\}$, where x_i represents the features of the ith observation. The goal of generative modelling is to train a model that learns the true underlying distribution of $\mathbf{X}$. The trained generative model can then be used to generate synthetic observations that closely resemble those of the original training set. In what follows, we provide a brief review of GANs and VAEs. The learning objectives for these deep generative models are modified later in Section 3 in order to model conditional distributions.

2.1. Generative adversarial networks

GANs, which were first introduced by I. Goodfellow et al. (2014), consist of two main components. The first is a generator $\mathcal{G}_\phi$, which takes as input a random noise variable $z \sim q_Z(z)$ and aims to generate a synthetic observation x' that closely resembles those belonging to $\mathbf{X}$. The second is a discriminator $\mathcal{J}_\theta$ that has the task of determining whether an observation is real or one that is generated by $\mathcal{G}_\phi$.

The learning objectives of $\mathcal{G}_\phi$ and $\mathcal{J}_\theta$ are thus in opposition to one another and can be written as a min–max problem, given by

$$\mathcal{L}_{\mathrm{GAN}}(x;\phi,\theta) = \min_{\phi}\max_{\theta}\Big(\mathbb{E}_{x\sim p_X(x)}\left[\log \mathcal{J}_\theta(x)\right] + \mathbb{E}_{z\sim q_Z(z)}\left[\log\left(1-\mathcal{J}_\theta(\mathcal{G}_\phi(z))\right)\right]\Big), \tag{1}$$

where $p_X(x)$ is the true, but unknown, distribution of $\mathbf{X}$. Once trained, we generate synthetic observations by first sampling $z \sim q_Z(z)$ and then passing this z to the trained $\mathcal{G}_\phi$ such that $x' = \mathcal{G}_\phi(z)$.

2.2. Variational autoencoders

VAEs are premised on x being generated by a random process involving a latent random variable z. Specifically, the process is such that an observation of z is first sampled from the prior distribution $p_\theta(z)$, which in turn is used to sample an observation of x from the conditional distribution $p_\theta(x|z)$.

The goal of the VAE is to obtain approximate maximum likelihood or maximum a posteriori estimates of the parameters θ in situations where both the marginal likelihood $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)\,\mathrm{d}z$ and the posterior $p_\theta(z|x)$ are intractable. It does so by utilizing the distribution $q_\phi(z|x)$ as an approximation to the intractable $p_\theta(z|x)$, and maximizing the variational lower bound for $p_\theta(x)$. The learning objective of the VAE is thus given by

$$\mathcal{L}_{\mathrm{VAE}}(x;\theta,\phi) = \max_{(\phi,\theta)}\Big(\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{KL}(q_\phi(z|x) \parallel p_\theta(z))\Big), \tag{2}$$

where $\mathbb{KL}(q(\cdot) \parallel p(\cdot))$ denotes the Kullback–Liebler divergence between two distributions $q(\cdot)$ and $p(\cdot)$. Once the VAE is trained, a synthetic observation, x', is generated by first sampling $z \sim p_\theta(z)$ and subsequently sampling x' from the trained probabilistic decoder $p_\theta(x|z)$.[1]

3. Deep conditional generative models for oversampling

In this section, we shift attention from unlabelled datasets to those which are labelled. Denote a labelled dataset by $(\mathbf{X},\mathbf{Y}) = \{(x_i,y_i), i = 1, 2, \dots, n\}$, where $y_i \in \{1, 2, \dots, L\}$ gives the class label of observation x_i. Moreover, let

$$\rho_\ell = \frac{1}{n}\sum_{i=1}^{n} 1_{(y_i=\ell)}, \quad \ell = 1, 2, \dots, L,$$

where $1_{(A)}$ is the indicator function of the event A. The dataset $(\mathbf{X},\mathbf{Y})$ is considered imbalanced if[2]

$$\min_\ell \rho_\ell \ll \max_\ell \rho_\ell.$$

It is important to recognize that it is possible for there to be more than one minority class that is under-represented in the dataset. We use $\mathbf{m}$ to denote the set of minority classes, and $\mathbf{M} = \{1, 2, \dots, L\}\backslash\mathbf{m}$ to denote the set of majority classes. In describing the type of imbalance, we adopt a convention similar to the one used in Buda et al. (2018). Specifically, we use two quantities to describe the severity of the imbalance. The first represents the fraction of minority classes $|\mathbf{m}|/L$, while the second gives the imbalance ratio ρ defined by

$$\rho = \frac{\max_\ell \rho_\ell}{\min_\ell \rho_\ell}.$$

In what follows, we review three kinds of conditional GAN models which are designed to learn class-dependent distributions. We also propose a conditional variant of the VAE. The main idea is to utilize these conditionalized generative models as a means to oversample imbalanced datasets by generating synthetic observations from the minority classes.

3.1. Oversampling with conditional GANs

The conditional GAN (CGAN) was first introduced by Mirza, and Osindero (2014) and extends the classical GAN to a conditional model where both the generator and discriminator are conditioned on the class label, y. For the generator, the input noise z and y are combined in a joint hidden representation. In similar fashion, x and y are passed

[1] It is important to realize that the VAE is a probabilistic autoencoder since both the encoder and decoder result in a distribution for z and x, respectively – one must sample from these distributions in order to obtain values for these two variables.

[2] The inequality relation "$\ll$" is applied loosely to mean "much" less than.

as inputs into the discriminator. The learning objective of a CGAN represents the same two-player minimax game as Eq. (1), but now both the discriminator and generator require y as input yielding

$$\mathcal{L}_{\text{CGAN}}(x, y; \phi, \theta) = \min_{\phi} \max_{\theta} \Big(\mathbb{E}_{x \sim p_X} [\log \mathcal{J}_\theta(x, y)] + \mathbb{E}_{z \sim q_Z} \big[\log\big(1 - \mathcal{J}_\theta(\mathcal{G}_\phi(z, y), y)\big) \big] \Big).$$

Once trained, to sample a class y observation, we first sample $z \sim q_Z(z)$ and then pass both z and y to the conditional generator to produce $x' = \mathcal{G}_\phi(z, y)$. The CGAN was applied to the imbalanced learning problem in Douzas and Bacao (2018), where the authors showed that, for binary classification problems, CGAN can outperform various traditional oversampling methods.

In addition to CGAN, there are two other variants of the traditional GAN that are also capable of producing class-dependent synthetic observations. The auxiliary classifier GAN (ACGAN) was introduced by Odena, Olah, and Shlens (2017) and its learning objective is comprised of two parts: the learning objective of the original GAN and another having to do with the correct classification of both real and synthetic observations. The balancing GAN (BAGAN) was introduced in Mariani et al. (2018) in order to generate minority class images. BAGAN utilizes a novel VAE–GAN dual structure, consisting of a three step training process. First, a VAE is trained on the observations, followed by the initialization of the GAN which is done by transferring the trained encoder and decoder of the VAE to various parts of the GAN, and finally adversarial training with the initialized GAN. For more details on ACGAN and BAGAN, we refer the interested reader to their original papers.

3.2. Oversampling with conditional VAEs

We next propose a variant of VAEs that aims to learn class-dependent distributions, referred to herein as the conditional VAE (CVAE). Conditionalizing the VAE merely requires a shift in the objective from learning the parameters (θ, ϕ) that maximize a lower bound on $p_\theta(x)$ (i.e., Eq. (2)) to instead learning (θ, ϕ) which maximize a similar lower bound for the conditional distribution $p_\theta(x|y)$. Indeed, the same line of mathematical reasoning used to establish Eq. (2) can be applied to achieve a variational lower bound for $p_\theta(x|y)$. Therefore, it follows that

$$\mathcal{L}_{\text{CVAE}}(x, y; \theta, \phi) = \max_{(\phi,\theta)} \Big(\mathbb{E}_{z \sim q_\phi(z|x,y)}[\log p_\theta(x|z, y)] - \mathbb{KL}\big(q_\phi(z|x, y) \parallel p_\theta(z|y)\big) \Big). \quad (3)$$

It should be noted that the exact same conditional learning objective was ultimately achieved by Kingma et al. (2014). However, their intent was to establish a variational lower bound for the joint distribution $p_\theta(x, y)$ in order to address the semi-supervised learning problem. This equivalence follows from the fact that $p_\theta(x, y) = p_\theta(x|y)p_\theta(y)$ and that the marginal $p_\theta(y)$ is not learned but instead assumed to be equal to the empirical probabilities of the class labels. It also bears mentioning that other versions of the CVAE were introduced by Sohn et al. (2015) and Lopez-Martin et al. (2017), but both of these differ from the one described above. Specifically, in Sohn et al. (2015), the authors utilized a conditional VAE framework to learn the structured output distribution of $p_\theta(y|x)$, whereas the loss in Eq. (3) is designed to learn the conditional distribution $p_\theta(x|y)$. On the other hand, Lopez-Martin et al. (2017) considered a conditionalized VAE that injects the labels at only the decoder stage, whereas we carry the label in both decoding and encoding phases. Observe that only injecting y at the decoder stage can be interpreted as making the simplifying assumption that z, conditioned on x, under the q_ϕ measure is independent of y such that $q_\phi(z|x, y) = q_\phi(z|x)$ for all y.

4. Numerical examples

We next assess the performance of the deep generative models discussed in the previous sections on two datasets, namely: MNIST (LeCun, Bottou, Bengio, & Haffner, 1998) and Fashion MNIST (Xiao, Rasul, & Vollgraf, 2017). Without modification, both MNIST and Fashion MNIST are well-balanced datasets. Therefore, for the purposes of the numerical results to follow, imbalanced versions of these datasets are created. Details on the exact process for skewing the MNIST and Fashion MNIST datasets are provided in Section 4.1.

For each of these datasets, an oversampling method is primarily judged based on the performance of a downstream classifier that is trained on a dataset comprised of the original imbalanced training set and a number of synthetic observations generated by the oversampling algorithm. In all examples, the number of synthetic observations used is exactly the amount required so that all classes are equally represented. However, we also consider the sample weight of the synthetic observations in the classifier loss function as a hyperparameter, which is tuned during downstream classifier training with cross validation. An alternative approach is to instead treat the number of synthetic observations itself as the hyperparameter. One method based on this idea was introduced in Li, Fong, Yuan, and Wong (2016), where the authors applied Particle-Swarm Optimization (PSO) techniques to determine the optimal number of synthetic observations. Using sample weight here as a hyperparameter is a pragmatic choice as PSO techniques are computationally intensive.

In all examples, we also compare the performance of the deep oversampling methods with two traditional oversampling methods: SMOTE (Chawla et al., 2002) and RANDOM oversampling (i.e., sampling with replacement in the minority classes). Finally, all computations are carried out on Amazon Web Services using an NVIDIA Tesla V100-SXM2-16GB Graphics Processing Unit, with the exception of the timing results reported in Table 9, which are performed on a MacBook Pro with a 1.4 GHz Quad-Core i5 processor.

4.1. Skewing MNIST and fashion MNIST

To artificially imbalance the MNIST dataset, we first consider three different sets of minority classes, $\mathbf{m}$, namely: $\{0, 1, 2, 3, 4, 5, 9\}$, $\{1, 2, 3, 4, 5\}$, and $\{3, 4, 5\}$. The reasoning behind this choice of sets stems from the fact that particular digits are, to the human eye, easier to separate than others. For example, there is more variance between 0 and 1 than there is between 3 and 8, or 4 and 9. Hence, in choosing these $\mathbf{m}$, similar digit pairs are separated to make classification more difficult for the machine learning algorithms. For example, if the digit 3 is in the minority while digit 8 is in the majority, an imbalanced training set is likely to produce a model that is biased towards predicting or generating 8's over 3's.

Next, for each $(\mathbf{m}, \mathbf{M})$ pair, we take the original MNIST training set of 60,000 images and randomly down-sample all of the majority classes so that each majority class is equally represented by 3,200 images. In similar fashion, we randomly down-sample all of the minority digits so that there is equal representation within the set of minority classes. We consider two different imbalance ratios to determine the number of minority digits that the down-sampling process yields. That is, $\rho = 80$ or 320, which is equivalent to down-sampling to 40 and 10 minority images, respectively.

Finally, this process of random selection of images is repeated for each $(\mathbf{m}, \mathbf{M}, \rho)$ triple five times. Therefore, for MNIST, we have a total of 30 (i.e., $3 \times 2 \times 5$) artificially imbalanced training sets. The original MNIST test set of 10,000 images is left untouched and used to assess the performance of the resultant downstream classifiers. Note that the original test set is well-balanced. An alternative approach is to also skew the test set, but doing so makes comparison across the various oversampling methods more difficult.

Table 1
Mean FID scores of minority classes for MNIST with $\rho = 320$.

Minority classes	Generative method	0	1	2	3	4	5	9
m = {0,1,2,3,4,5,9}	SMOTE	57.5	43.9	**63.2**	49.1	55.9	57.6	55.3
	ACGAN	106.5	116	131.7	88.1	94.6	110.1	96.2
	BAGAN	191.6	151.9	190	157.1	114.1	145.3	136.7
	CGAN	285.5	160.2	249.7	189.5	204	253.2	203.9
	CVAE	104.9	136.6	101.3	88.3	84.2	95.3	71.7
	R-ACGAN	152.8	184.7	149.3	122.8	142	142.3	141.1
	R-CGAN	54.8	64.7	74.2	49.9	61.4	61.1	60.7
	R-CVAE	**42.3**	**32**	64.1	**33.3**	**50.5**	**45.5**	**37.5**
m = {1,2,3,4,5}	SMOTE		45.1	61.9	55.5	55.8	54.2	
	ACGAN		108.1	96.1	75.2	80.9	69.2	
	BAGAN		162.3	211.1	166.7	144.2	162.1	
	CGAN		196.9	206.8	175.6	178.8	189.9	
	CVAE		136.1	82.3	78.6	68	73.8	
	R-ACGAN		45.7	63.2	49.8	57.1	55.5	
	R-CGAN		50.7	66.3	47	54.4	50.7	
	R-CVAE		**33.8**	**57.5**	**38.9**	**45.7**	**39.3**	
m = {3,4,5}	SMOTE				48.7	53.8	58.8	
	ACGAN				110.8	113.7	72.6	
	BAGAN				199.1	157.2	228	
	CGAN				220.9	162	249	
	CVAE				82	69.5	84.4	
	R-ACGAN				42.2	50.2	63.5	
	R-CGAN				45.4	58	56.5	
	R-CVAE				**38.4**	**52.2**	**47.5**	

Table 2
Mean FID scores of minority classes for MNIST with $\rho = 80$.

Minority classes	Generative method	0	1	2	3	4	5	9
m = {0,1,2,3,4,5,9}	SMOTE	**34.6**	25.8	**36.3**	**28.9**	**29.2**	**34**	**28**
	ACGAN	76.2	52.9	167.3	71.9	113.1	103.6	97.8
	BAGAN	93.8	92.1	132.7	86	94.9	97.7	83.5
	CGAN	232.2	161.3	201	146.4	191.8	179.6	181.3
	CVAE	67.4	75.3	94.5	61.8	62.3	76.1	49.7
	R-ACGAN	217	232.3	211.3	172.1	204.8	201.9	202.3
	R-CGAN	113.3	144.1	104	84.4	104.9	100.9	112.1
	R-CVAE	40.3	**22.9**	66.8	33.7	47.2	47.4	34.2
m = {1,2,3,4,5}	SMOTE		**25.5**	**35.3**	**28.1**	**30.7**	**33.1**	
	ACGAN		68.1	131.6	84.2	79.5	75.7	
	BAGAN		84.3	151.7	93	82	94.4	
	CGAN		134.8	188.6	147.3	155	160.1	
	CVAE		92.2	84.3	62.6	57.5	66.3	
	R-ACGAN		238.3	202.3	158.3	184.3	187.7	
	R-CGAN		137.2	130	89.9	119	98.8	
	R-CVAE		27.5	71.6	39.3	50.4	47.8	
m = {3,4,5}	SMOTE				**26.9**	**29.3**	**31.9**	
	ACGAN				59.3	84.2	69.3	
	BAGAN				119	97.5	134.7	
	CGAN				185.5	227.6	184.6	
	CVAE				57.8	50.6	62.9	
	R-ACGAN				135.2	172	144.4	
	R-CGAN				75.3	82.5	78.4	
	R-CVAE				33.8	49.8	44.7	

Similar to the MNIST dataset, Fashion MNIST consists of 60,000 training and 10,000 test images of clothing articles. We apply the same process described above for MNIST to create 30 artificially imbalanced training sets for Fashion MNIST. Indeed, the choice of **m** and ρ values is guided by the same principle for Fashion MNIST as it is for MNIST. That is, for example, images of t-shirts are separated from images of shirts, while images of sneakers are separated from images of sandals.

4.2. Quality of generated observations

When dealing with image datasets, there is another way to assess the performance of the generative models. The Frechet inception distance (FID) was proposed by Heusel, Ramsauer, Unterthiner, Nessler, and Hochreiter (2017) and represents a measure to assess the quality of a set of generated images by comparing them to a set of real ones. Specifically, FID measures the distance between empirical distributions of generated and real images, under the assumption that the data points follow a Gaussian distribution in the feature space. Generated images that achieve lower FID scores are viewed as being of higher quality.

Using the 30 artificially imbalanced MNIST training sets, several synthetic observations are generated with trained ACGAN, BAGAN, CGAN, CVAE, and SMOTE models. We then compute the FID scores between the generated images and the original MNIST test set. Tables 1 and 2 report the mean FID scores per minority class achieved by each generative model across the five training sets associated with the triples (**m**, **M**, $\rho = 320$) and (**m**, **M**, $\rho = 80$), respectively. Similarly, the mean FID scores for the Fashion MNIST dataset are shown in Tables 3 and 4.

It is important to realize that the deep generative models must also be trained on the imbalanced training set. In certain settings, the low number of minority class observations can hinder the learning of

Table 3
Mean FID scores of minority classes for Fashion MNIST with $\rho = 320$.

Minority classes	Generative method	t-shirt 0	Trouser 1	Pullover 2	Dress 3	Coat 4	Sandal 5	Ankle boot 9
m = {0,1,2,3,4,5,9}	SMOTE	90.4	53.4	80	86.3	86.5	87	66.8
	ACGAN	204.2	160.6	251.2	175	235.2	174.7	177.3
	BAGAN	275.7	308.5	303.7	240.7	271.2	251.5	290.2
	CGAN	233.3	281.7	242	239.8	249.8	258	256
	CVAE	103.8	151.1	115.6	124.1	93.3	107	143.2
	R-ACGAN	346.5	357	352.2	342.7	347.1	292.5	341.7
	R-CGAN	92.3	52.5	76.4	91	83.2	92.9	70.7
	R-CVAE	**74.2**	**43.1**	**61.2**	**69**	**62.1**	**68.5**	**53**
m = {1,2,3,4,5}	SMOTE		54.3	86.1	85.9	82.1	87.1	
	ACGAN		190.8	275.4	181.8	248.7	185.4	
	BAGAN		325	284.4	232	259.7	250.8	
	CGAN		282.2	284.8	218.8	240.2	225.7	
	CVAE		158	112.5	107.1	91.9	108.6	
	R-ACGAN		73.6	118.2	120.6	113.9	121.4	
	R-CGAN		56.7	88.7	87.5	84.9	97.3	
	R-CVAE		**42.5**	**71.8**	**65**	**65.4**	**67.3**	
m = {3,4,5}	SMOTE				91.8	87.1	85.6	
	ACGAN				150.7	207.9	226.2	
	BAGAN				264.9	261	288.4	
	CGAN				190.7	193.7	243.5	
	CVAE				92.9	73.6	114.7	
	R-ACGAN				96.5	102.6	107.3	
	R-CGAN				97.4	96.2	94.8	
	R-CVAE				**72.1**	**68.1**	**68.8**	

Table 4
Mean FID scores of minority classes for Fashion MNIST with $\rho = 80$.

Minority classes	Generative method	t-shirt 0	Trouser 1	Pullover 2	Dress 3	Coat 4	Sandal 5	Ankle boot 9
m = {0,1,2,3,4,5,9}	SMOTE	**44.6**	**28.3**	**48.1**	**47.2**	**45.6**	**49.9**	**36.6**
	ACGAN	224.7	133.9	319.9	249	298.3	200.3	250.9
	BAGAN	241.6	230.8	259.5	178.7	225.3	200.3	245.2
	CGAN	262.9	244.1	277.4	230.3	267.2	247.3	290.2
	CVAE	83	67.4	99.3	94.2	80.7	93.1	120
	R-ACGAN	344	312.2	317.2	304.1	300.5	267.3	274.5
	R-CGAN	170.9	136	183.1	166.3	174.8	157.6	154.8
	R-CVAE	56.5	28.4	60.8	51.3	57.4	60.8	58.6
m = {1,2,3,4,5}	SMOTE		28.9	**48.9**	**47.3**	**46.8**	**51.7**	
	ACGAN		262	369.6	309.2	352.6	243	
	BAGAN		253.4	257.5	209.6	203.2	204.4	
	CGAN		300.1	294	231.8	261.4	246.1	
	CVAE		75.8	93.2	88.6	79.6	101.6	
	R-ACGAN		268.9	317.5	283.8	285.3	249.6	
	R-CGAN		108.3	158.9	137.1	135.6	126.9	
	R-CVAE		**27.6**	59.3	52.8	60.3	65.2	
m = {3,4,5}	SMOTE				**48.6**	**45.7**	**46.1**	
	ACGAN				333.1	345.1	261.3	
	BAGAN				185.7	221.6	197.1	
	CGAN				244.3	218.1	240.4	
	CVAE				82.9	64	97.7	
	R-ACGAN				263.4	300.3	235.1	
	R-CGAN				60.9	66.4	53.8	
	R-CVAE				54.3	56.7	61.2	

good feature space structures for deep generative models. One possible remedy for this problem is to first randomly oversample the minority observations to balance the training set and subsequently train the deep generative model on this balanced dataset. This technique is applied to each of the considered deep generative models, with the exception BAGAN,[3] in all experiments. The resulting FID scores are also displayed in Tables 1–4, prefixed by R. For example, R-CVAE represents random oversampling to balance the training set followed by training a CVAE model on the balanced dataset. Note that for both CGAN and CVAE, randomly oversampling the imbalanced training set first leads to higher quality generated images. Interestingly, the mean FID scores for R-ACGAN are not always lower than that of ACGAN. We believe that this is likely due to the presence of the image classification component in the learning objective of ACGAN.

A possible alternative to random oversampling is training multiple, class-specific generative models as is done, for instance, in Pham, Dutt, Pellerin, and Quénot (2019). Such an approach alleviates the challenge of training being dominated by majority class examples. However, as noted in Santurkar, Schmidt, and Madry (2018), constructing separate models significantly reduces the data available for learning and forgoes the possibility of leveraging inter-class relationships. That is, in monolithic training, such as that considered here, inter-class commonalities and distinctions serve to improve model generalizability overall. Evidence of this is seen numerically in Tables 5–8, discussed in detail below, where the classifiers built on top of, for example, CVAE-based

[3] BAGAN is significantly more computationally intensive as compared to the other deep generative models considered here, due to it requiring the training of both a VAE and GAN component.

Table 5
Downstream classifier results for MNIST with $\rho = 320$.

Type	$\mathbf{m} = \{0, 1, 2, 3, 4, 5, 9\}$			$\mathbf{M} = \{6, 7, 8\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.946	0.398	0.537	0.455	0.992	0.613	0.798	0.571	0.555	−15.138
SMOTE	0.947	0.516	0.656	0.508	0.990	0.661	0.815	0.654	0.654	–
RANDOM	0.943	0.522	0.657	0.513	0.991	0.666	0.814	0.659	0.657	0.459
ACGAN	0.884	0.423	0.553	0.479	0.990	0.635	0.762	0.589	0.574	−12.232
BAGAN	0.942	0.452	0.595	0.469	0.992	0.629	0.799	0.609	0.602	−7.951
CGAN	0.921	0.411	0.549	0.459	0.990	0.618	0.782	0.581	0.566	−13.456
CVAE	0.814	0.568	0.662	0.591	0.964	**0.725**	0.745	0.683	0.677	3.517
R-ACGAN	0.878	0.422	0.555	0.473	0.991	0.631	0.754	0.589	0.574	−12.232
R-CGAN	0.940	0.515	0.651	0.505	0.990	0.660	0.809	0.654	0.651	−0.459
R-CVAE*	0.945	0.561	**0.694**	0.534	0.990	0.685	0.821	0.686	**0.688**	5.199
Type	$\mathbf{m} = \{1, 2, 3, 4, 5\}$			$\mathbf{M} = (0, 6, 7, 8, 9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.982	0.339	0.478	0.639	0.985	0.762	0.810	0.657	0.614	−13.521
SMOTE	0.969	0.483	0.627	0.690	0.986	0.801	0.829	0.729	0.710	–
RANDOM	0.969	0.506	0.648	0.697	0.985	0.807	0.833	0.740	0.723	1.831
ACGAN	0.918	0.446	0.586	0.685	0.983	0.797	0.801	0.712	0.689	−2.958
BAGAN	0.981	0.370	0.513	0.651	0.986	0.771	0.816	0.673	0.637	−10.282
CGAN	0.970	0.327	0.464	0.644	0.985	0.764	0.807	0.652	0.610	−14.084
CVAE*	0.831	0.559	**0.662**	0.755	0.968	**0.841**	0.792	0.761	**0.750**	5.634
R-ACGAN	0.948	0.462	0.599	0.683	0.984	0.796	0.815	0.717	0.692	−2.535
R-CGAN	0.965	0.481	0.623	0.692	0.985	0.802	0.828	0.728	0.708	−0.282
R-CVAE	0.963	0.512	0.650	0.704	0.984	0.811	0.833	0.743	0.726	2.254
Type	$\mathbf{m} = \{3, 4, 5\}$			$\mathbf{M} = \{0, 1, 2, 6, 7, 8, 9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.980	0.231	0.362	0.799	0.983	0.870	0.851	0.757	0.715	−8.333
SMOTE	0.972	0.378	0.533	0.826	0.983	0.889	0.868	0.801	0.780	–
RANDOM	0.973	0.386	0.541	0.827	0.983	0.890	0.869	0.803	0.784	0.513
ACGAN	0.856	0.364	0.500	0.830	0.983	0.892	0.836	0.797	0.773	−0.897
BAGAN	0.968	0.249	0.387	0.804	0.983	0.873	0.850	0.762	0.725	−7.051
CGAN	0.922	0.272	0.403	0.808	0.980	0.876	0.840	0.767	0.731	−6.282
CVAE*	0.865	0.553	**0.668**	0.866	0.973	**0.912**	0.864	0.846	**0.837**	7.308
R-ACGAN	0.932	0.294	0.434	0.811	0.982	0.878	0.845	0.775	0.742	−4.872
R-CGAN	0.966	0.387	0.541	0.826	0.983	0.889	0.866	0.804	0.783	0.385
R-CVAE	0.962	0.412	0.567	0.835	0.983	0.895	0.871	0.811	0.794	1.795

balancing methods see marked improvements in both minority and majority class performance. Nevertheless, a comparative study of these two approaches in the context of oversampling is an interesting avenue for future work.

With regard to the best performing generative model, observe that, for both MNIST and Fashion MNIST datasets, R-CVAE yields the lowest mean FID scores when $\rho = 320$. However, for $\rho = 80$, while R-CVAE produces the lowest FID scores amongst all deep generative methods, SMOTE achieves the lowest overall FID scores. It is worth mentioning, however, that SMOTE generates synthetic images as interpolations of in-class neighbouring samples. The topology of the observation space is approximately preserved in the feature space generated by the inception layer. In other words, as long as the real neighbouring images are similar enough to one another, measured, for example, by the Frobenius norm of the differences of the two images, interpolations of these images would also not deviate too much from the distribution in the feature space, and thus we would expect lower FID scores. Nonetheless, the in-class interpolations of SMOTE lead to shadowy generated images, as seen in Figs. 1a and 2a, compared with those of the R-CVAE model. Finally, each CVAE produces lower mean FID scores than their GAN counterparts. We emphasize that, in the balanced MNIST case, GANs produce lower FID scores than VAEs (Dai, & Wipf, 2019, Table 1). Thus, the results reported in Tables 1–4 suggest that the CVAEs are more robust to imbalanced training sets than conditional GANs.

4.3. Downstream classifier performance

As previously mentioned, the various oversampling methods are primarily evaluated on their ability to improve the performance of a downstream classifier. For both datasets, MNIST and Fashion MNIST, we employ a simple multi-layer perceptron (MLP) for downstream classification. As is customary for MLPs, and other neural network architectures, the network weights are randomly initialized before training. As a result, a certain amount of randomness exists in the performance of the downstream classification. Thus, for each of the 30 artificially imbalanced training sets of MNIST and Fashion MNIST, augmented by a given oversampling method, three separate downstream MLP models are trained. This implies that each of the five training sets associated with the triple $(\mathbf{m}, \mathbf{M}, \rho)$ have three collections of test-set performance metrics. Each of these metrics is averaged across all 15 instances to produce a single set of performance measures associated with a given $(\mathbf{m}, \mathbf{M}, \rho)$ triple.

Tables 5 and 6 display such averages to three decimal places for every oversampling method in the MNIST example. Similarly, we report these averages to three decimal places for the Fashion MNIST example in Tables 7 and 8. Asterisks indicate methods exhibiting the best F_1 score for the minority classes in each experiment. Each table also contains the averages of precision and recall across three sets of images: $\mathbf{m}$, $\mathbf{M}$, and all classes (i.e., overall). In addition to demonstrating a downstream classifier's average performance with each of the oversampling techniques, the performance of a classifier trained on the imbalanced datasets is shown to serve as a baseline for performance when label skew is left unaddressed. These results are denoted by "imbalanced" rows in the respective tables. Finally, each table details the relative percentage increase, or decrease, in a classifiers overall F_1 score compared to that achieved by the classifiers built using SMOTE for oversampling. This is calculated as

$$\frac{F_{1,\alpha} - F_{1,\mathrm{SMOTE}}}{F_{1,\mathrm{SMOTE}}} \cdot 100,$$

Table 6
Downstream classifier results for MNIST with $\rho = 80$.

Type	$\mathbf{m} = \{0,1,2,3,4,5,9\}$			$\mathbf{M} = \{6,7,8\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.953	0.726	0.819	0.650	0.990	0.780	0.861	0.803	0.805	−5.738
SMOTE	0.951	0.797	0.864	0.726	0.987	0.833	0.883	0.852	0.854	–
RANDOM*	0.953	0.801	**0.868**	0.727	0.987	0.834	0.885	0.855	**0.857**	0.351
ACGAN	0.944	0.740	0.823	0.670	0.988	0.794	0.861	0.812	0.812	−4.918
BAGAN	0.944	0.714	0.808	0.644	0.988	0.775	0.853	0.794	0.797	−6.674
CGAN	0.947	0.727	0.816	0.659	0.989	0.785	0.859	0.802	0.804	−5.855
CVAE	0.908	0.791	0.843	0.752	0.969	**0.844**	0.861	0.841	0.841	−1.522
R-ACGAN	0.949	0.727	0.818	0.658	0.990	0.785	0.861	0.804	0.806	−5.621
R-CGAN	0.947	0.731	0.820	0.661	0.989	0.788	0.860	0.806	0.809	−5.269
R-CVAE	0.946	0.788	0.856	0.719	0.983	0.827	0.877	0.844	0.846	−0.937
Type	$\mathbf{m} = \{1,2,3,4,5\}$			$\mathbf{M} = \{0,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.974	0.689	0.800	0.776	0.985	0.864	0.875	0.834	0.830	−4.817
SMOTE	0.972	0.769	0.854	0.822	0.984	0.893	0.897	0.874	0.872	–
RANDOM*	0.972	0.782	**0.863**	0.830	0.984	0.898	0.901	0.881	**0.879**	0.803
ACGAN	0.962	0.734	0.828	0.805	0.984	0.882	0.883	0.856	0.853	−2.179
BAGAN	0.972	0.694	0.803	0.776	0.984	0.864	0.874	0.835	0.831	−4.702
CGAN	0.965	0.680	0.790	0.775	0.985	0.863	0.869	0.829	0.824	−5.505
CVAE	0.927	0.783	0.845	0.851	0.975	**0.906**	0.888	0.876	0.874	0.229
R-ACGAN	0.972	0.700	0.807	0.784	0.985	0.868	0.877	0.839	0.835	−4.243
R-CGAN	0.970	0.713	0.817	0.791	0.985	0.873	0.880	0.846	0.843	−3.326
R-CVAE	0.966	0.765	0.850	0.821	0.983	0.892	0.893	0.871	0.869	−0.344
Type	$\mathbf{m} = \{3,4,5\}$			$\mathbf{M} = \{0,1,2,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.972	0.627	0.759	0.873	0.983	0.921	0.901	0.876	0.872	−3.646
SMOTE	0.970	0.728	0.830	0.900	0.983	0.938	0.920	0.906	0.905	–
RANDOM	0.974	0.721	0.826	0.897	0.983	0.936	0.919	0.904	0.902	−0.331
ACGAN	0.958	0.682	0.795	0.890	0.983	0.932	0.909	0.892	0.890	−1.657
BAGAN	0.970	0.623	0.755	0.873	0.983	0.921	0.901	0.875	0.870	−3.867
CGAN	0.969	0.618	0.750	0.873	0.983	0.921	0.900	0.873	0.869	−3.978
CVAE*	0.934	0.760	**0.836**	0.910	0.974	**0.940**	0.916	0.909	**0.908**	0.442
R-ACGAN	0.965	0.657	0.779	0.881	0.983	0.927	0.905	0.885	0.882	−2.541
R-CGAN	0.969	0.661	0.782	0.882	0.983	0.927	0.907	0.886	0.883	−2.431
R-CVAE	0.963	0.706	0.813	0.893	0.982	0.934	0.913	0.899	0.896	−0.994

for an oversampling method α. Relative increases are highlighted in green.

As part of each of the numerical experiments, stratified 5-fold validation studies are conducted to determine the best value for the number of neighbours used by SMOTE for interpolation based on the overall F_1 score for a downstream MLP classifier. These studies are conducted for each of the 30 existing imbalanced training sets using possible neighbour values of $\{3,4,5,6,7\}$. However, the results of such parameter studies did not consistently yield better performance than simply fixing the number of neighbours to 5. It is believed that this phenomenon is due to the limited number of minority class examples in the training set. With a limited number of examples available for validation, the F_1 scores are less reliable estimates of eventual test-set performance and thereby may lead to sub-optimal parameter choices on average compared with the fixed value. As such, the SMOTE metrics reported in Tables 5–8 correspond to this fixed value. In addition, the tuning of SMOTE requires training multiple downstream classifiers in this setting and is, therefore, more time consuming. See Table 9 for time comparisons.

In Tables 5–8, first note that in all 12 cases, using synthetic samples generated by SMOTE, RANDOM, CVAE, and R-CVAE all lead to an improvement in the average F_1 score for **m** compared with that of a classifier trained on the imbalanced dataset. For ACGAN and R-ACGAN, there are only three occasions where there was no improvement over the imbalanced case, whereas for BAGAN there are five such cases. The worst performing deep generative model was CGAN, whose samples did not lead to an improvement in the downstream classifier performance in eight of the 12 experiments for the minority classes. The results of R-CGAN are significantly better for minority classes than those of CGAN and exhibit improvement in all experiments. However, the F_1 scores remain much lower than those of CVAE and R-CVAE. Notably, SMOTE, CVAE, and R-CVAE also result in better F_1 scores for the majority classes, **M**, and overall in all experiments suggesting that these oversampling methods lead to better overall classifier learning and the benefits are not isolated to minority class performance. Finally, observe that much of the gains in F_1 score for minority classes is attributable to significant increases in minority class recall, while maintaining a similar or greater level of precision.

Considering the best performing generative methods in the experiments, observe that:

- CVAE achieves the best F_1 score for **m** in 3 of the 12 cases;
- R-CVAE achieves the best F_1 score for **m** in 3 of the 12 cases;
- SMOTE achieves the best F_1 score for **m** in 4 of the 12 cases;
- RANDOM achieves the best F_1 score for **m** in 2 of the 12 cases.

Therefore, generally speaking, the CVAE methodology, composed of CVAE and R-CVAE, accounts for six of the 12 best F_1 scores for **m**. Further, if overall F_1 score is considered, these results improve such that CVAE techniques account for seven of the top 12 scores. This indicates that the CVAE-based methods for oversampling strongly influence classifier performance for all classes. Indeed, as mentioned above, majority class metrics are noticeably improved when using these models, in some cases enough so to yield superior overall F_1 metrics even when minority class results are not best-in-class. Moreover, both the CVAE and R-CVAE models outperform all other deep generative models in every experiment, based on minority class and overall F_1 scores, and by a wide margin in many cases. Such performance strongly favours the CVAE-based approach as the best deep generative model

Table 7
Downstream classifier results for Fashion MNIST with $\rho = 320$.

Type	$\mathbf{m} = \{0,1,2,3,4,5,9\}$			$\mathbf{M} = \{6,7,8\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.617	0.211	0.275	0.486	0.984	0.607	0.578	0.443	0.375	−39.123
SMOTE	0.817	0.497	0.591	0.560	0.916	0.675	0.740	0.623	0.616	–
RANDOM	0.821	0.481	0.574	0.569	0.921	**0.679**	0.745	0.613	0.605	−1.786
ACGAN	0.618	0.211	0.279	0.495	0.983	0.615	0.581	0.442	0.380	−38.312
BAGAN	0.631	0.193	0.260	0.490	0.986	0.611	0.588	0.431	0.365	−40.747
CGAN	0.449	0.110	0.147	0.446	0.977	0.575	0.448	0.370	0.275	−55.357
CVAE	0.755	0.368	0.464	0.531	0.950	0.650	0.688	0.543	0.520	−15.584
R-ACGAN	0.720	0.218	0.281	0.477	0.983	0.603	0.647	0.447	0.378	−38.636
R-CGAN	0.868	0.361	0.466	0.528	0.977	0.651	0.766	0.546	0.522	−15.260
R-CVAE*	0.791	0.522	**0.604**	0.568	0.889	0.677	0.724	0.632	**0.626**	1.623
Type	$\mathbf{m} = \{1,2,3,4,5\}$			$\mathbf{M} = \{0,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.633	0.194	0.240	0.597	0.915	0.703	0.615	0.555	0.472	−25.434
SMOTE	0.824	0.436	0.526	0.649	0.883	0.739	0.736	0.660	0.633	–
RANDOM	0.832	0.438	0.529	0.648	0.885	0.740	0.740	0.661	0.634	0.158
ACGAN	0.739	0.221	0.280	0.598	0.913	0.705	0.668	0.567	0.492	−22.275
BAGAN	0.708	0.208	0.262	0.597	0.914	0.704	0.653	0.561	0.483	−23.697
CGAN	0.471	0.116	0.154	0.583	0.914	0.692	0.527	0.515	0.423	−33.175
CVAE	0.787	0.424	0.510	0.674	0.895	**0.755**	0.731	0.660	0.633	0.000
R-ACGAN	0.780	0.277	0.356	0.624	0.914	0.725	0.702	0.595	0.540	−14.692
R-CGAN	0.873	0.331	0.429	0.632	0.914	0.731	0.752	0.622	0.580	−8.373
R-CVAE*	0.814	0.470	**0.552**	0.667	0.875	0.746	0.740	0.673	**0.649**	2.528
Type	$\mathbf{m} = \{3,4,5\}$			$\mathbf{M} = \{0,1,2,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.521	0.016	0.029	0.662	0.898	0.753	0.620	0.634	0.536	−18.417
SMOTE*	0.857	0.287	**0.392**	0.703	0.870	0.770	0.749	0.695	0.657	–
RANDOM	0.920	0.174	0.275	0.685	0.897	0.772	0.756	0.680	0.623	−5.175
ACGAN	0.707	0.051	0.091	0.667	0.901	0.760	0.679	0.646	0.559	−14.916
BAGAN	0.718	0.025	0.047	0.659	0.900	0.753	0.677	0.638	0.541	−17.656
CGAN	0.642	0.035	0.064	0.662	0.901	0.755	0.656	0.641	0.548	−16.591
CVAE	0.856	0.237	0.343	0.705	0.891	**0.781**	0.750	0.695	0.650	−1.065
R-ACGAN	0.902	0.105	0.181	0.677	0.899	0.764	0.744	0.661	0.589	−10.350
R-CGAN	0.929	0.172	0.275	0.692	0.901	0.776	0.763	0.683	0.625	−4.871
R-CVAE	0.870	0.276	0.386	0.705	0.876	0.774	0.754	0.696	**0.658**	0.158

for oversampling. It should be noted that, in practice, the choice of whether to use random oversampling in training a CVAE model for oversampling requires observing results on a validation set in order to determine which training method is likely to yield the best classifier performance.

It is interesting to observe that, as the number of minority classes decreases, such that $|\mathbf{m}|/L$ becomes smaller, the performance of the downstream classifier on the minority classes also decreases. This phenomenon is likely due to the presence of more and more majority class examples, which increases the difficulty of learning good minority-class representations. Nevertheless, the CVAE oversampling methodology achieves the best overall F_1 score for three out of the four cases in which $\mathbf{m} = \{3,4,5\}$.

Of the six experiments where CVAE-based methods yield superior minority-class F_1 performance, five are associated with the case that $\rho = 320$. That is, with larger class imbalance, such methods outperform SMOTE and RANDOM oversampling, with the exception of a single experiment. When considering the overall F_1 score, a CVAE model performs best in all cases where $\rho = 320$. In addition, for $\rho = 80$, CVAE yields the best overall F_1 score in one of the experiments. In the remaining cases, oversampling with the CVAE models yields competitive results compared to SMOTE and are the only deep generative models to consistently do so. These results indicate that the CVAE methodology produces strong results in all circumstances, but particularly distances itself from simpler methods like SMOTE and RANDOM oversampling in cases of more severe class imbalance.

The largest observed relative improvements over SMOTE in overall F_1 score for CVAE and R-CVAE are 7.308% and 5.199%, respectively, both of which occur when $\rho = 320$. On the other hand, the largest margin by which SMOTE outperforms the two approaches is -15.584% for CVAE and -1.560% for R-CVAE. This minimum for CVAE is actually found when $\rho = 320$ for Fashion-MNIST, where many of the deep generative methods struggle significantly. Based on the results in Tables 7 and 8, it appears that random oversampling prior to generative model training is particularly important for this dataset, as the only deep model to produce better results than SMOTE is R-CVAE. Further to this point, SMOTE is actually outperformed by the vanilla random oversampling method in two of the six experiments contained in those tables.

One drawback of all deep generative methods considered in this paper is that they are computationally intensive compared with RANDOM and SMOTE oversampling. While CVAE is less expensive than some of the other deep methods that it outperforms, it still requires significantly more resources than its closest competitor in SMOTE. Table 9 displays the average time, in seconds, required to perform the training set balancing for MNIST and Fashion MNIST in the various settings considered above for CVAE and SMOTE. The CVAE timing results include training the model for 10 epochs, where the loss is already beginning to plateau, while the times for Tuned SMOTE include the training of downstream MLP classifiers used to estimate the best neighbours parameter. Note that there is a fairly large cost to tuning SMOTE with downstream classifier performance. In some cases, this cost brings the overall time to balance the training set in line with the CVAE oversampling times in the table.

The CVAE models are trained for substantially longer than 10 epochs in the numerical experiments above and, as such, still require considerably more overall time. Therefore, in practice, such computational cost must be taken into account when deciding on a method to use for

Table 8
Downstream classifier results for Fashion MNIST with $\rho = 80$.

Type	$\mathbf{m} = \{0,1,2,3,4,5,9\}$			$\mathbf{M} = \{6,7,8\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.885	0.396	0.485	0.561	0.977	0.672	0.788	0.570	0.541	−27.577
SMOTE*	0.821	0.700	**0.747**	0.676	0.851	0.746	0.778	0.745	**0.747**	–
RANDOM	0.834	0.680	0.738	0.665	0.879	**0.749**	0.783	0.739	0.742	−0.669
ACGAN	0.848	0.409	0.498	0.562	0.978	0.676	0.762	0.580	0.552	−26.104
BAGAN	0.860	0.373	0.464	0.545	0.979	0.663	0.766	0.554	0.524	−29.853
CGAN	0.774	0.386	0.471	0.556	0.977	0.671	0.708	0.564	0.531	−28.916
CVAE	0.844	0.582	0.675	0.636	0.946	0.738	0.782	0.691	0.694	−7.095
R-ACGAN	0.893	0.419	0.508	0.571	0.970	0.681	0.796	0.584	0.560	−25.033
R-CGAN	0.880	0.435	0.533	0.576	0.972	0.687	0.789	0.596	0.579	−22.490
R-CVAE	0.812	0.700	0.740	0.681	0.840	0.745	0.773	0.742	0.741	−0.803
Type	$\mathbf{m} = \{1,2,3,4,5\}$			$\mathbf{M} = \{0,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.915	0.394	0.496	0.659	0.913	0.750	0.787	0.653	0.623	−18.986
SMOTE*	0.831	0.687	**0.737**	0.758	0.858	**0.801**	0.794	0.772	**0.769**	–
RANDOM	0.865	0.648	0.730	0.741	0.885	0.801	0.803	0.767	0.766	−0.3901
ACGAN	0.909	0.375	0.475	0.660	0.915	0.749	0.785	0.645	0.612	−20.416
BAGAN	0.898	0.358	0.453	0.655	0.914	0.746	0.776	0.636	0.599	−22.107
CGAN	0.772	0.325	0.411	0.651	0.913	0.741	0.711	0.619	0.576	−25.098
CVAE	0.863	0.588	0.681	0.727	0.893	0.792	0.795	0.740	0.736	−4.292
R-ACGAN	0.877	0.386	0.484	0.662	0.915	0.753	0.770	0.651	0.618	−19.636
R-CGAN	0.897	0.416	0.528	0.662	0.909	0.751	0.779	0.663	0.639	−16.905
R-CVAE	0.837	0.656	0.723	0.738	0.863	0.790	0.787	0.760	0.757	−1.560
Type	$\mathbf{m} = \{3,4,5\}$			$\mathbf{M} = \{0,1,2,6,7,8,9\}$			Overall			
Method	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1	Lift %
Imbalanced	0.921	0.273	0.396	0.717	0.903	0.792	0.778	0.714	0.673	−13.273
SMOTE*	0.831	0.654	**0.712**	0.792	0.835	0.803	0.804	0.781	0.776	–
RANDOM	0.880	0.602	0.704	0.783	0.874	**0.822**	0.812	0.792	**0.786**	1.289
ACGAN	0.941	0.266	0.381	0.713	0.898	0.787	0.782	0.709	0.665	−14.304
BAGAN	0.930	0.277	0.396	0.717	0.902	0.792	0.781	0.715	0.673	−13.273
CGAN	0.885	0.240	0.353	0.710	0.902	0.788	0.762	0.703	0.658	−15.206
CVAE	0.895	0.482	0.614	0.760	0.890	0.815	0.800	0.768	0.755	−2.706
R-ACGAN	0.956	0.247	0.363	0.708	0.903	0.788	0.782	0.706	0.660	−14.948
R-CGAN	0.933	0.403	0.543	0.744	0.900	0.808	0.800	0.751	0.729	−6.057
R-CVAE	0.825	0.624	0.690	0.783	0.838	0.801	0.795	0.774	0.768	−1.031

Table 9
Computation time in seconds for SMOTE and CVAE oversampling.

MNIST						
Method	$\mathbf{m} = \{0,1,2,3,4,5,9\}$		$\mathbf{m} = \{1,2,3,4,5\}$		$\mathbf{m} = \{3,4,5\}$	
	$\rho = 320$	$\rho = 80$	$\rho = 320$	$\rho = 80$	$\rho = 320$	$\rho = 80$
SMOTE	0.436	0.381	0.488	0.318	0.343	0.319
Tuned SMOTE	300	351	320	358	339	363
CVAE	324	319	532	490	709	640
Fashion MNIST						
Method	$\mathbf{m} = \{0,1,2,3,4,5,9\}$		$\mathbf{m} = \{1,2,3,4,5\}$		$\mathbf{m} = \{3,4,5\}$	
	$\rho = 320$	$\rho = 80$	$\rho = 320$	$\rho = 80$	$\rho = 320$	$\rho = 80$
SMOTE	0.370	0.390	0.379	0.316	0.367	0.320
Tuned SMOTE	482	472	482	471	485	469
CVAE	547	513	756	748	1066	1067

oversampling. The results above indicate that, when class imbalance is severe, oversampling with CVAE models provides substantive benefits if the computation costs are not prohibitive. In more balanced cases, such models perform well but do not yet consistently surpass SMOTE. This motivates further research into more sophisticated CVAE models in order to continue to improve oversampling performance.

5. Conclusions

In this paper, we considered the handling of the class imbalance problem through oversampling via conditional variants of VAEs and GANs. In the numerical experiments, we observed that random oversampling prior to training the deep conditional model improves both FID scores as well as, in many cases, the performance of a downstream classifier. The FID analysis in this paper also suggests that CVAEs proposed herein are more robust to imbalanced datasets than their conditional GAN counterparts – a result that is reversed when applying GANs and VAEs to the original, well-balanced MNIST and Fashion MNIST datasets.

Most importantly, the experiments suggest that the CVAE methodology is the best deep conditional generative model for oversampling to correct class imbalance. Moreover, the technique consistently outperforms SMOTE and RANDOM oversampling methods in cases of

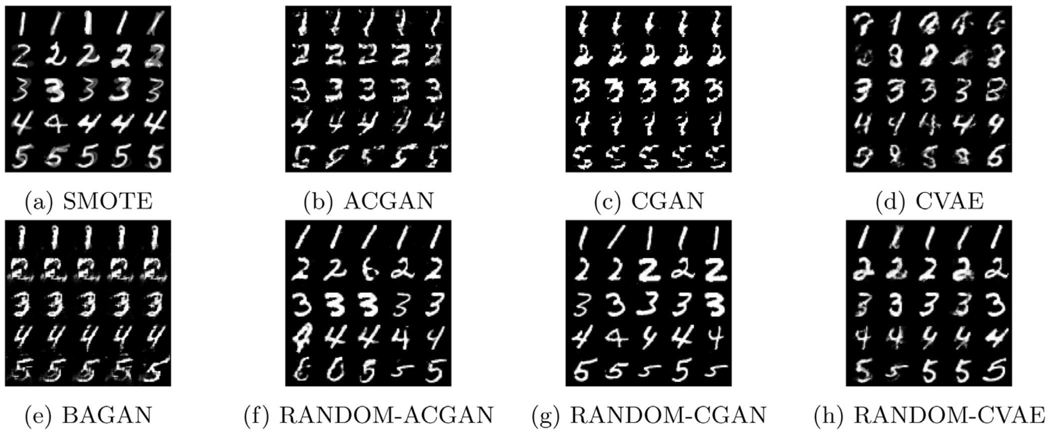

Fig. 1. Examples of synthetic images from imbalanced MNIST with minority classes $\mathbf{m} = (1, 2, 3, 4, 5)$ and $\rho = 320$.

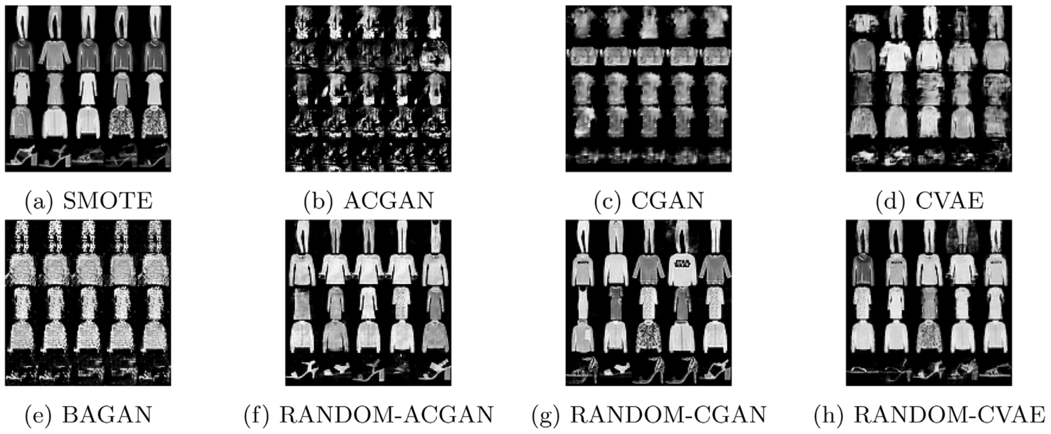

Fig. 2. Examples of synthetic images from imbalanced Fashion MNIST with minority classes $\mathbf{m} = (1, 2, 3, 4, 5)$ and $\rho = 320$.

Table 10
ACGAN specifications.

Group	Layer	Output size	Maps	Kernel size	Stride	Batch-norm.	Activation
Generator							
	Input = $[z, y]$	62 + 10	–	–	–	–	–
	Fully connected	1024	–	–	–	Yes	ReLU
	Fully connected	6272	–	–	–	Yes	ReLU
	Reshape	7×7	128	–	–	–	–
	Deconvolution	14×14	64	4×4	2	Yes	ReLU
	Deconvolution	28×28	1	4×4	2	No	Sigmoid
Discriminator							
	Input	28×28	1	–	–	–	–
	Convolution	14×14	64	4×4	2	No	Leaky ReLU
	Convolution	7×7	128	4×4	2	Yes	Leaky ReLU
	Fully connected	1024	–	–	–	Yes	Leaky ReLU
	Fully connected	1	–	–	–	No	Sigmoid
Auxiliary classifier							
	Input	1024	–	–	–	–	–
	Fully connected	128	–	–	–	Yes	Leaky ReLU
	Fully connected	10	–	–	–	No	SoftMax
Generator optimizer	AdamOptimizer (learning_rate = 0.0002, beta1 = 0.5)						
Discriminator optimizer	AdamOptimizer (learning_rate = 0.001, beta1 = 0.5)						
Classifier optimizer	AdamOptimizer (learning_rate = 0.001, beta1 = 0.5)						
Leaky ReLU slope	0.2						
Epochs	60						
Epochs (R-ACGAN)	60						
$q_Z(z)$	Uniform(−1,1)						
Backend	Tensorflow						

Table 11
CGAN specifications.

Group	Layer	Output size	Maps	Kernel size	Stride	Batch-norm.	Activation
Generator							
	Input = $[z, y]$	$62 + 10$	–	–	–	–	–
	Fully connected	1024	–	–	–	Yes	ReLU
	Fully connected	6272	–	–	–	Yes	ReLU
	Reshape	7×7	128	–	–	–	–
	Deconvolution	14×14	64	4×4	2	Yes	ReLU
	Deconvolution	28×28	1	4×4	2	No	Sigmoid
Discriminator							
	Input	28×28	1	–	–	–	–
	Convolution	14×14	64	4×4	2	No	Leaky ReLU
	Convolution	7×7	128	4×4	2	Yes	Leaky ReLU
	Fully connected	1024	–	–	–	Yes	Leaky ReLU
	Fully connected	1	–	–	–	No	Sigmoid
Generator optimizer	AdamOptimizer (learning_rate = 0.0002, beta1 = 0.5)						
Discriminator optimizer	AdamOptimizer (learning_rate = 0.001, beta1 = 0.5)						
Leaky ReLU slope	0.2						
Epochs	30						
Epochs (R-CGAN)	60						
$q_Z(z)$	Uniform(−1,1)						
Backend	Tensorflow						

Table 12
BAGAN specifications.

Generator optimizer	AdamOptimizer (learning_rate = 0.00005, beta1 = 0.5)
Discriminator optimizer	AdamOptimizer (learning_rate = 0.00005, beta1 = 0.5)
Reconstruction optimizer	AdamOptimizer (learning_rate = 0.00005, beta1 = 0.5)
Epochs	30
Backend	Keras

severe imbalance (i.e., higher ρ values and smaller $|\mathbf{m}|/L$). The class imbalance problem is one of extreme importance, and we hope that these numerical examples can serve as a benchmark for future research in this area (see the Appendix for the webpage link from which the skewed datasets can be downloaded).

With respect to future work, we aim to apply these deep generative models to non-image datasets. The numerical experiments suggest that in situations where oversampling improves downstream performance, the CVAE methodology should be considered as a viable option for producing synthetic observations. Additionally, there have been many enhancements made to the VAE (e.g., see van den Berg, Hasenclever, Tomczak, & Welling, 2018; Burda, Grosse, & Salakhutdinov, 2016; Dai & Wipf, 2019). Of particular interest is the 2-Stage VAE introduced by Dai and Wipf (2019), which involves the training of two VAEs. The first is trained on the dataset $\mathbf{X}$, and the second is trained on the encoded samples, $\mathbf{Z} = \{z\}_{i=1}^{N}$, that are generated via the probabilistic encoder of the first VAE. This two-stage process aids in the learning of the ground-truth measure of $\mathbf{X}$ and leads to FID scores that are lower than that of the traditional VAE. Conditionalizing the two-Stage VAE is accomplished by simply training two CVAEs in each of the two stages. We are also interested in developing conditional versions of Wasserstein autoencoders (Kolouri, Martin, & Rohde, 2019; Tolstikhin, Bousquet, Gelly, & Scholkopf, 2018), which have attracted much attention in recent years.

CRediT authorship contribution statement

Val Andrei Fajardo: Conceptualization, Investigation, Software, Writing. **David Findlay:** Investigation, Software. **Charu Jaiswal:** Investigation, Software, Writing. **Xinshang Yin:** Data curation, Software, Validation. **Roshanak Houmanfar:** Investigation, Software. **Honglei Xie:** Investigation, Software. **Jiaxi Liang:** Investigation. **Xichen She:** Software Writing. **D.B. Emerson:** Revision, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 13
CVAE specifications.

Group	Layer	Output size	Maps	Kernel size	Stride	Batch-norm.	Activation
Encoder							
	Input = $[x, y]$	28×28	10	–	–	–	–
	Convolution	14×14	64	4×4	2	No	Leaky ReLU
	Convolution	7×7	128	4×4	2	Yes	Leaky ReLU
	Fully connected	1024	–	–	–	Yes	Leaky ReLU
	Fully connected	62+62	–	–	–	No	Linear
Decoder							
	Input = $[z, y]$	$62 + 10$	–	–	–	–	–
	Fully connected	1024	–	–	–	Yes	ReLU
	Fully connected	6272	–	–	–	Yes	ReLU
	Reshape	7×7	128	–	–	–	–
	Deconvolution	14×14	64	4×4	2	Yes	ReLU
	Deconvolution	28×28	1	4×4	2	No	Sigmoid
ELBO optimizer	AdamOptimizer (learning_rate = 0.001, beta1 = 0.5)						
Leaky ReLU slope	0.2						
Epochs	300						
Epochs (R-CVAE)	125						
$q_Z(z)$	MultivariateNormal($\mathbf{0}_{62}$, $\mathbf{I}_{62\times 62}$)						
Backend	Tensorflow						

Table 14
Downstream MLP specifications.

Layer	Output size	Dropout	Activation
Input	784	–	–
Fully connected	512	0.2	ReLU
Fully connected	512	0.2	ReLU
Fully connected	10	–	Softmax
Cross-entropy optimizer	RMSprop()		
Epochs	20		
Batch size	128		
K-fold cross validation	2		
Hyperparameter	Synthetic sample weight		
Grid search space	Numpy.linspace(1,50,10)		
Grid search score	Weighted average minority class F_1 score		
Backend	Keras		

Appendix

To promote the reproducibility of the above analyses, we have made all of the skewed training sets of MNIST and Fashion MNIST publicly available for download. Interested readers can access the data through this link: http://anonymous-research-public.s3-website.ca-central-1.amazonaws.com. Additionally, we provide the model specifications for each of the deep generative models used in the analyses in Tables 10–14. Note that the same architectures are used for both MNIST and Fashion MNIST examples.

The implementations of ACGAN, CGAN, and CVAE are built from Wu (2019). BAGAN, on the other hand, is implemented by using the implementation found in Mariani, and Martinelli (2019); and so, we omit the architecture details of BAGAN here and provide only the specifications which we altered in the experiments, see Table 12. Finally, for SMOTE, we use the implementation from the Python imblearn module.

References

van den Berg, R., Hasenclever, L., Tomczak, J. M., & Welling, M. (2018). Sylvester normalizing flows for variational inference. In *34th Conference on uncertainty in artificial intelligence* (pp. 393–402). Association For Uncertainty in Artificial Intelligence (AUAI).

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks, 106*, 249–259.

Burda, Y., Grosse, R., & Salakhutdinov, R. (2016). Importance weighted autoencoders. In *4th International conference on learning representations*.

Chawla, N., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Dai, B., & Wipf, D. (2019). Diagnosing and enhancing VAE models. In *International conference on learning representations*.

Douzas, G., & Bacao, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications, 91*, 464–471.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications, 73*, 220–239.

He, H., & Garcia, E. A. (2008). Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9), 1263–1284.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in neural information processing systems* (pp. 6626–6637).

I. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 5967–5976).

Kingma, D., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems* (pp. 3581–3589).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations*.

Kolouri, S., Martin, C. E., & Rohde, G. K. (2019). Sliced-Wasserstein autoencoder: An embarrassingly simple generative model. In *International conference on learning representations*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Li, J., Fong, S., Yuan, M., & Wong, R. K. (2016). Adaptive multi-objective swarm crossover optimization for imbalanced data classification. In *International conference on advanced data mining and applications* (pp. 374–390). Springer.

Liang, D., Krishnan, R. B., Hoffman, M. D., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World wide web conference* (pp. 689–698).

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IOT. *Sensors, 17*(9), 1967.

Louizos, C., Swersky, K., Li, Y., Welling, M., & Zemel, R. (2016). The variational fair autoencoder. In *4th International conference on learning representations*.

Maloof, M. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II* (vol. 2) (pp. 2–1).

Mariani, G., & Martinelli, S. (2019). *BAGAN*. IBM Research, GitHub repository.

Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., & Malossi, C. (2018). BAGAN: Data augmentation with balancing GAN. arXiv preprint arXiv:1803.09655.

McCarthy, K., Zabar, B., & Weiss, G. (2005). Does cost-sensitive learning beat sampling for classifying rare classes? In *Proceedings of the 1st international workshop on utility-based data mining* (pp. 69–77). ACM.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th international conference on machine learning (vol. 70)* (pp. 2642–2651). JMLR. org.

Pham, T., Dutt, A., Pellerin, D., & Quénot, G. (2019). Classifier training from a generative model. In *17th International conference on content-based multimedia indexing* (pp. 1–6).

Richard, M., & Lippmann, R. P. (1991). Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation, 3*(4), 461–483.

S. Lawrence, S., Burns, I., Back, A., Tsoi, A. C., & Giles, C. L. (1998). Neural network classification and prior class probabilities. In *Neural networks: Tricks of the trade* (pp. 299–313). Springer.

Samangouei, P., Kabkab, M., & Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International conference on learning representations*.

Santurkar, S., Schmidt, L., & Madry, A. (2018). A classification–based study of covariate shift in GAN distributions. In *35th International conference on machine 35th international conference on machine learning* (pp. 4480–4489).

Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (pp. 3483–3491).

Tolstikhin, I., Bousquet, O., Gelly, S., & Scholkopf, B. (2018). Wasserstein auto-encoders. In *International conference on learning representations*.

Wu, G. (2019). tensorflow-GANs. GitHub repository.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.